Average Path Length of Binary Decision Diagrams

Jon T. Butler, *Fellow*, *IEEE*, Tsutomu Sasao, *Fellow*, *IEEE*, and Munehiro Matsuura

Abstract—The traditional problem in binary decision diagrams (BDDs) has been to minimize the number of nodes since this reduces the memory needed to store the BDD. Recently, a new problem has emerged: minimizing the average path length (APL). APL is a measure of the time needed to evaluate the function by applying a sequence of variable values. It is of special significance when BDDs are used in simulation and design verification. A main result of this paper is that the APL for benchmark functions is typically much smaller than for random functions. That is, for the set of all functions, we show that the average APL is close to the maximum path length, whereas benchmark functions show a remarkably small APL. Surprisingly, however, typical functions do *not* achieve the absolute maximum APL. We show that the parity functions are unique in having that distinction. We show that the APL of a BDD can vary considerably with variable ordering. We derive the APL for various functions, including the AND, OR, threshold, Achilles' heel, and certain arithmetic functions. We show that the unate cascade functions uniquely achieve the absolute minimum APL.

Index Terms—Binary decision diagrams, BDD, average path length, APL, worst-case path length.

1 Introduction

Considerable research has been devoted to the use of binary decision diagrams (BDDs) in logic design, dating back 40 years to Lee's [1] original paper. Most of this research followed from the seminal paper by Bryant [2], who showed that a reduced ordered BDD (ROBDD) is a canonical representation of a logic function. The focus of the latter paper and many papers [2], [3], [4], [5] that followed has been on minimizing the number of nodes. This is inspired by the memory needed to store the BDD, which can be large for practical circuits.

However, a different cost measure is also important. For each assignment of values to the variables of a function $f(X)$, there is a path from the root node to one of two terminal nodes, representing a value of $f(X)$. Summing this path length over all 2^n assignments and dividing by 2^n yields the *average path length* or *APL* of $f(X)$. In general, the APL depends on the order of the variables and we denote the minimum APL over all possible orderings as $APL_{f(X)}$. In logic simulation, the function realized is verified by applying (typically many) test vectors [7], [8], [9]. A BDD with a small APL can be evaluated quickly and is important in logic simulation applications. There are at least three papers that describe the minimization of the APL of BDDs [9], [10], [11]. Iguchi et al. [12] compare the APL for four different decision diagrams representing multioutput functions. Minimizing the APL of multivalued functions is discussed in [13].

- *J.T. Butler is with the Department of Electrical and Computer Engineering, Naval Postgraduate School, Code EC/Bu, Monterey, CA 93943-5121. E-mail: jbutler@nps.navy.mil.*
- *T. Sasao and M. Matsuura are with the Department of Computer Science and Electronics, Kyushu Institute of Technology, Iizuka-shi, Fukuoka-ken, 820-8502, Japan. E-mail: {sasao, matsuura}@cse.kyutech.ac.jp.*

Manuscript received 25 Aug. 2003; revised 18 June 2004; accepted 28 Jan. 2005; published online 15 July 2005.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-0138-0803.

BDDs have been used in the design of pass transistor logic circuits [28], [29], [30], [31], which have the advantage of low power dissipation. In this case, there is the prospect of choosing an ordering with a small APL as this yields smaller delay. Also, in a logic synthesis involving functional decomposition, the paths in the BDD representing the function create don't care values from missing variables. In this case, minimizing the path length has the effect of increasing don't care values [32], which, in turn, reduces circuit complexity.

APL has been important in other contexts. For example, Bell [14] has considered its use in decision trees for pattern recognition systems. In this application, one seeks to answer the fewest questions needed, on the average, to identify the object. APL is useful in the analysis of algorithms, where decision trees have been used to determine lower bounds on the complexity of sorting algorithms [15]. Applications also occur in databases [16]. Qin and Malik [17] consider APL in decision trees for the decoding logic for microprocessor instructions. For more information on research prior to 1982, the reader is referred to Moret [18]. For information on more recent research, the reader is referred to Murthy [19]. We address the following question:

Does the APL of a BDD of a logic function vary significantly enough with the variable ordering to merit a close examination?

In Fig. 1, we show the BDD of the carry-out of a binary adder. Fig. 1a shows the circuit, which consists of b full-adders connected in a ripple-carry configuration. It is assumed there is a carry-in variable, c_{in}.

Fig. 1b shows the BDD for this function, where the variables are in descending order (most significant bits at the top), and Fig. 1c shows the BDD in which the variables are in ascending order (least significant bits at the top). Fig. 2 shows the distribution of path lengths for the two orderings for a $b = 16$ bit adder. The pattern associated with the most significant bit at the top is a sawtooth because there are no paths of odd length. In this distribution, most paths are short, resulting in an APL of 4.0. The paths associated with the least significant bits at the top are long, resulting in a

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
JUN 2004		

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Average Path Length of Binary Decision Diagrams	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Naval Postgraduate School,Department of Electrical and Computer Engineering,Monterey,CA,93943**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The traditional problem in binary decision diagrams (BDDs) has been to minimize the number of nodes since this reduces the memory needed to store the BDD. Recently, a new problem has emerged: minimizing the average path length (APL). APL is a measure of the time needed to evaluate the function by applying a sequence of variable values. It is of special significance when BDDs are used in simulation and design verification. A main result of this paper is that the APL for benchmark functions is typically much smaller than for random functions. That is, for the set of all functions, we show that the average APL is close to the maximum path length, whereas benchmark functions show a remarkably small APL. Surprisingly, however, typical functions do not achieve the absolute maximum APL. We show that the parity functions are unique in having that distinction. We show that the APL of a BDD can vary considerably with variable ordering. We derive the APL for various functions, including the AND, OR, threshold, Achilles? heel, and certain arithmetic functions. We show that the unate cascade functions uniquely achieve the absolute minimum APL.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**		**13**	

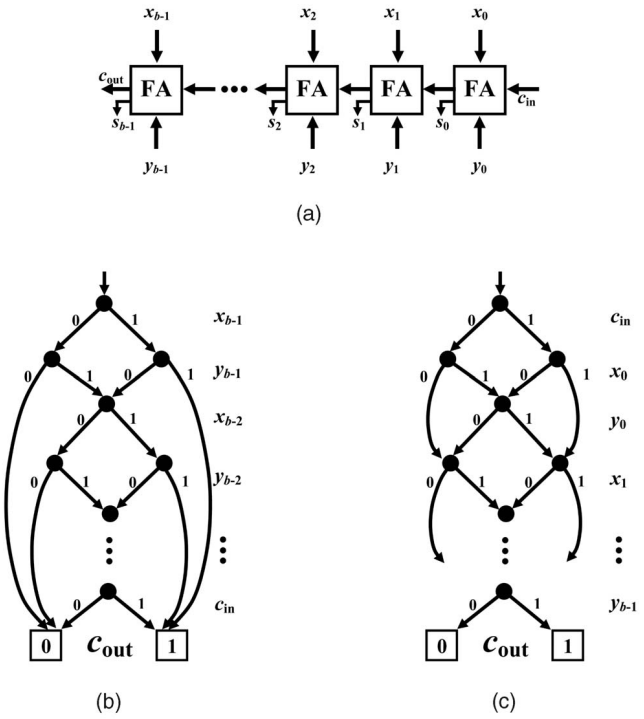

Fig. 1. Two BDDs for the carry-out function of a b-bit ripple carry adder. (a) Carry-out c_{out} of b-bit ripple carry adder circuit. (b) BDD for c_{out} with MSB at top. (c) BDD for c_{out} with LSB at top.

large APL, 25.0. Both orderings yield the same number of nodes. Thus, the APL of the carry-out function is strongly dependent on the variable ordering.

In this paper, we focus on reduced ordered binary decision diagrams (ROBDD), as proposed by Bryant [2]. As such, the order of the variables along all paths from the root node to a terminal node is the same (corresponding to the O of ROBDD). Further, each subfunction, as represented by an assignment of values to variables beginning at the root node, is represented by a unique node (corresponding to the R of ROBDD). As is common, we use BDD to abbreviate ROBDD.

In this paper, we do not discuss methods for minimizing the APL of BDDs. There are a number of approaches for doing this. One is to enumerate all orderings and to choose the one that produces the smallest APL. A more efficient approach is branch and bound [11], which can be applied to functions with up to 25 variables. Both methods find the exact minimum. For functions on more variables, heuristic methods must be used. Two heuristics include a window permutation approach [10] and a dynamic variable reordering method [11], [33].

This paper is organized as follows: In Section 2, we discuss specific functions. As far as we know, this is the first analysis of important functions with respect to the APL of their BDD. In Section 3, we analyze the average of the APL for classes of functions, including all functions and all symmetric functions. We show that these averages are near the maximum values. We show, however, that the set of symmetric threshold functions does not share this characteristic. In Section 4, we show that benchmark functions have a relatively small APL. Section 5 discusses conclusions. The reader interested in just the results has only to read the main part. For the reader interested in the details, we have provided proofs in the Appendix.

2 The APL of Specific Functions

2.1 AND/OR Function

The BDD of an n-variable AND function, shown in Fig. 3a consists of a single path from the root node to the terminal node labeled 1. For any node, there is an edge to the terminal node labeled 0. Therefore, we expect the APL to be small.

Theorem 1.

$$APL_{AND(n)} = 2 - \frac{1}{2^{n-1}}.$$

Proof. See the Appendix. □

This has been reported by Breithart and Gal [20]. We extend this by deriving also the distribution of path lengths for the AND function (see the Appendix).

Since the BDD of the OR function is isomorphic to that of the AND function, $APL_{AND(n)} = APL_{OR(n)}$. The AND and OR functions belong to a special class of functions.

2.2 Unate Cascade Functions

Definition 1 [21]. *f is a unate cascade function if f can be represented as*

$$f(x_1, x_2, \ldots x_n) = x_1^* \diamondsuit_1 (x_2^* \diamondsuit_2 (\ldots (x_{n-1}^* \diamondsuit_{n-1} x_n^*)) \ldots),$$

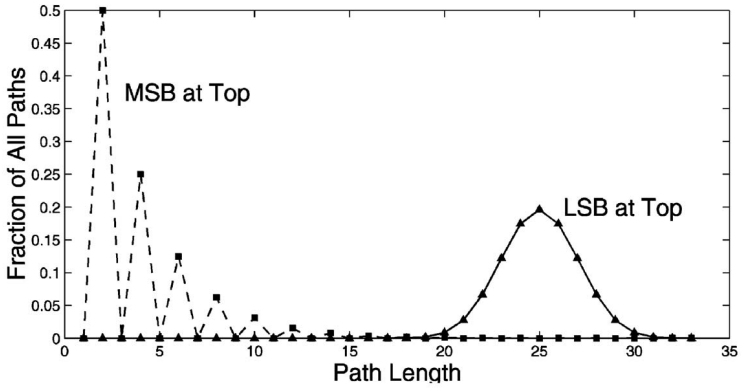

Fig. 2. Distributions of path lengths for the carry-out function.

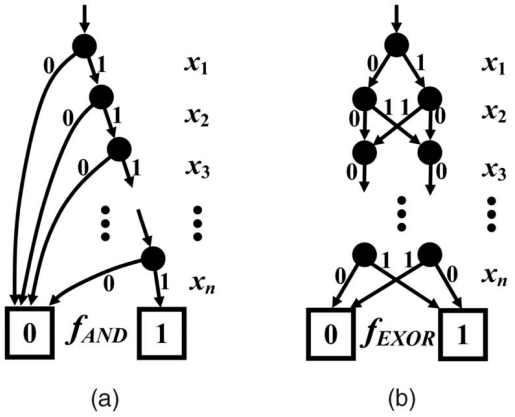

Fig. 3. BDDs for the (a) AND and (b) EXOR functions.

where x_i^* is either x_i or $\bar{x}_i$ and $\diamondsuit_i$ is either the OR ($\vee$) or AND ($\wedge$) operator.

The AND and OR functions are unate cascade functions. The BDD of a unate cascade function has one edge from every node to one of the two terminal nodes, except the bottom node, where two edges are incident to the terminal nodes. We have

Theorem 2. *The unate cascade functions are uniquely those functions whose BDDs have the smallest APL ($2 - \frac{1}{2^{n-1}}$) among all functions that depend on n-variables.*

Proof. See the Appendix. □

Having identified exactly those functions whose BDD has the smallest APL, we now consider which functions have the largest APL.

2.3 Parity Functions

Fig. 3b shows the BDD of the Exclusive OR function. All paths have length n, a result of the fact that every variable in every assignment of values is needed to determine the function value. These statements also apply to the complement of the Exclusive OR function. Collectively, these two functions form the *parity functions*. Since all path lengths are n, the APL is n.

Theorem 3. *The parity functions are uniquely those functions whose BDDs have the largest APL (n) among all functions that depend on n variables.*

Proof. See the Appendix. □

Theorems 2 and 3 completely characterize the functions with the smallest and the largest APL. We now establish properties of functions whose APL falls between these extremes, beginning with functions whose BDDs are similar to the parity functions. From Moret [18], we have

Definition 2. *A variable x_i in function $f(X)$ is indispensable iff for every assignment $\overrightarrow{a} \in B^{n-1}$ of values to $X - \{x_i\}$, $f(\overrightarrow{a})|_{x_i=0} \neq f(\overrightarrow{a})|_{x_i=1}$, where $B = \{0, 1\}$.*

For example, in the function

$$f(x_1, x_2, x_3) = x_1 \oplus x_2 \oplus x_3 x_4,$$

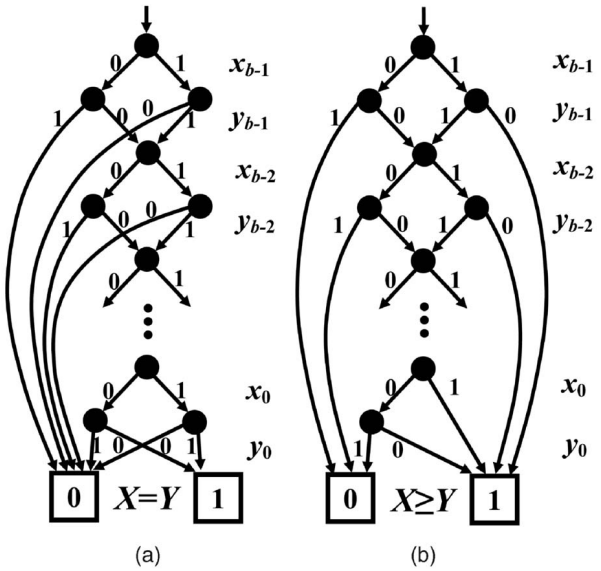

Fig. 4. BDDs for two comparison functions. (a) Equal-to function. (b) Greater-than-or-equal-to function.

x_1 and x_2 are indispensable, but x_3 and x_4 are not. The descriptor "indispensable" is due to the fact that it is always necessary to specify the value of an indispensable variable in order to determine the value of the function. From this, we can conclude

Theorem 4. *Let $f(X)$ be a function with N indispensable variables. Then, all paths in any BDD of $f(X)$ have a length of at least N, and $APL_{f(X)} \geq N$.*

For a given assignment $\overrightarrow{a}$ of values to $X - \{x_i\}$, $f(X)|_{x_i=0} = \bar{f}(X)|_{x_i=1}$. It follows that $f(X)|_{x_i=0} = \bar{f}(X)|_{x_i=1}$ and $f(X) = f(X)|_{x_i=0} \oplus x_i$. Thus, we can state

Theorem 5. *Let $f(X)$ be a function with N indispensable variables, y_1, y_2, ..., and y_N. Then,*

$$f(X) = g(X - \{y_1, y_2, ..., y_N\}) \oplus y_1 \oplus y_2 \oplus \ldots \oplus y_N.$$

Theorem 5 shows that the functions with the most indispensable variables are the two parity functions. It also shows that there are no functions dependent on n variables that have $n-1$ dispensable variables. There are functions with $n-2$ indispensable variables, e.g., $f(x_1, x_2, x_3) = x_1 \oplus x_2 \oplus x_3 x_4$.

2.4 Comparison and Carry-Out Functions

We examine two comparison functions, the Equal-to and the Greater-than-or-equal-to functions, as well as the carry out function. Let $X = \{x_{b-1}, x_{b-2}, \ldots, x_0\}$ and $Y = \{y_{b-1}, y_{b-2}, \ldots, y_0\}$ be the input variables that represent a standard binary number. That is, $x = x_{b-1}2^{b-1} + x_{b-2}2^{b-2} + \ldots + x_0 2^0$ and $y = y_{b-1}2^{b-1} + y_{b-2}2^{b-2} + \ldots + y_0 2^0$ are the standard binary representations corresponding to logic variable sets X and Y, respectively. Then, the Equal-to function $f_{x=y}(X, Y) = 1$ iff $x = y$ and the Greater-than-or-equal-to function $f_{x \geq y}(X, Y) = 1$ iff $x \geq y$. Fig. 4 shows the BDDs for these functions.

The Equal-to function is 1 iff the pair of ith significant bits are identical for all i. Thus, its BDD structure is

TABLE 1
APL for Carry-Out and Comparison Functions

Function	APL
Carry-out	$4 - 3/2^b$
Equal-to	$4 - 4/2^b$
Greater-than-or-equal-to	$4 - 5/2^b$

unaffected by a reordering of the pairs. For the Greater-than-or-equal-to function, the smallest APL occurs when the MSB is at the top, as is the case for the carry-out function. Fig. 4 shows its structure when the MSB is at the top. We can state

Theorem 6. *The smallest APL in a BDD for the Greater-than-or-equal-to function is $4 - \frac{5}{2^b}$ and is achieved when variables are ordered in ascending significance.*

Proof. See the Appendix. □

Note that the BDD structure for the carry-out, as shown in Fig. 1b, and the comparison functions, as shown in Fig. 4, are similar. This suggests that their APL is similar. Table 1 shows the APL for each of these functions. It is a constant 4 less a function of b, the number of bits, that approaches 0 as b increases. Thus, in the limit, all three BDDs have an APL of 4. Note that the (small) difference in the APL values is explained entirely by the structure at the bottom of the BDDs.

2.5 Symmetric Threshold Functions

A *(totally) symmetric function* is unchanged by any permutation of its input variables. For example, the AND, OR, and parity functions are symmetric. A *symmetric threshold function* is a symmetric function that is 1 iff t or more of its variables are 1, for $0 \leq t \leq n+1$. For example, the AND and OR functions are symmetric threshold functions, where $t = n$ and $t = 1$, respectively. However, the Exclusive OR function is not a symmetric threshold function, although it is a symmetric function. There are $n+2$ symmetric threshold functions on n variables.

Fig. 5 shows the distribution of path lengths for 17 symmetric threshold functions on 33 variables. The

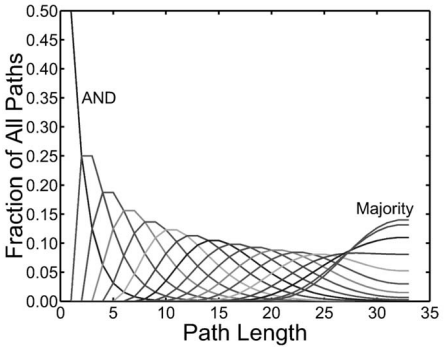

Fig. 5. Distribution of path lengths for 33-variable symmetric threshold functions for t from 33 (AND) to 17 (Majority).

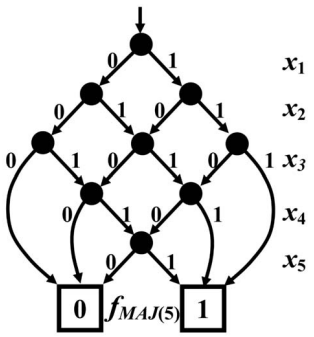

Fig. 6. BDD of the 5-variable majority function.

AND function shown on the extreme left corresponds to $t = 33$, while the majority function (which is 1 iff a majority of the variables are 1) shown on the extreme right corresponds to $t = 17$. Approximately one-half of the symmetric threshold functions are shown; all except the majority function distributions occur as pairs. For example, the distribution of path lengths associated with the AND function, where $t = 33$, also represents the distribution of path lengths for the OR function, where $t = 1$. For the APL of a symmetric threshold function, including trivial functions $f = 0$ and $f = 1$, we have

Theorem 7.

$$APL_{S\_THRES(n,t)} = 2k - \sum_{j=1}^{k} \frac{\binom{n-j}{k-j}}{2^{n-j}} j,$$

where $k = \min\{t, n-t+1\}$.

Proof. See the Appendix. □

When n is large, we can write

Corollary 1.

$$APL_{S\_THRES(n,t)} \sim 2\min\{t, n-t+1\},$$

where $A(n) \sim B(n)$ *means* $\lim_{n\to\infty} \frac{A(n)}{B(n)} = 1$.

2.6 Majority Function

When n is odd and t and n are related by $n = 2t-1$, the symmetric threshold function is the majority function. One of the two functions realized by the circuit labeled FA in Fig.1a is a 3-variable majority function: It is 1 iff two or three variables are 1. Fig. 6 shows the BDD of the majority function on five variables, which resembles a square. We have

Theorem 8.

$$APL_{MAJ(n)} = n - \frac{n+1}{2^n}\binom{n}{\frac{n-1}{2}} + 1, \quad (1)$$

where n is odd.

Proof. See the Appendix. □

The second term contains the factor $\binom{n}{\frac{n-1}{2}}$, which is $\frac{n!}{\frac{n-1}{2}!\frac{n+1}{2}!}$. The factorials can be approximated using Stirling's approximation, yielding

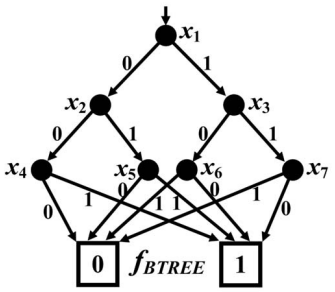

Fig. 7. BDD of the 7-variable BTREE function.

Corollary 2.

$$APL_{MAJ(n)} \sim n - \sqrt{\frac{2n}{\pi}} + 1. \tag{2}$$

We could also write $APL_{MAJ(n)} \sim n$, but the inclusion of terms $-\sqrt{\frac{2n}{\pi}}+1$ in (2) improves the approximation's precision for small values of n. For $n = 3$ and $n = 15$, the exact expression and the approximation of (2) differ by 4.5 percent and 0.4 percent, respectively.

Note that, for a general symmetric threshold function whose APL is approximated in Corollary 1, we assume the threshold t stays constant as $n \to \infty$. For the APL of the Majority function given in Corollary 2, we assume that $n = 2t - 1$ always holds so that t increases as n increases.

2.7 BTREE Function

We now consider the BTREE logic function that is characterized by the structure of its BDD. Namely, it is in the form of a binary tree, where each node is labeled by a unique variable. Therefore, a BTREE function on n variables has n nonterminal nodes. We restrict BTREEs to be balanced: Namely, no path in a BTREE has a length of more than 1 longer than any other path. When $n+1$ is a power of 2, the BTREE is a complete balanced tree and *every* path has length $\log_2(n+1)$. Fig. 7 shows a BDD of a BTREE function on seven variables. The BTREE is important because its *maximum* path length is the smallest among all n variable functions. For a discussion of the importance of the longest path length in BDDs, the reader is referred to [6]. For general BTREE functions, we can state

Theorem 9.

$$APL_{BTREE(n)} = \lfloor \log_2(n+1) \rfloor + \frac{(n+1) - 2^{\lfloor \log_2(n+1) \rfloor}}{\lfloor \log_2(n+1) \rfloor}.$$

Proof. See the Appendix. □

The BTREE function is interesting because of the following:

Theorem 10. *The longest path length in a BDD of an n-variable function is bounded below by $\lceil \log_2(n+1) \rceil$. The BTREE function achieves this lower bound.*

2.8 Achilles' Heel Function

The Achilles' heel function [22] is often used to show the effect of variable order on the number of nodes in a BDD. In this section, we derive the APL of a BDD for the Achilles' heel function for the *good* order, which produces a BDD with n nodes, and the *poor* order which produces a BDD with $2^{\frac{n}{2}+1} - 2$ nodes [2].

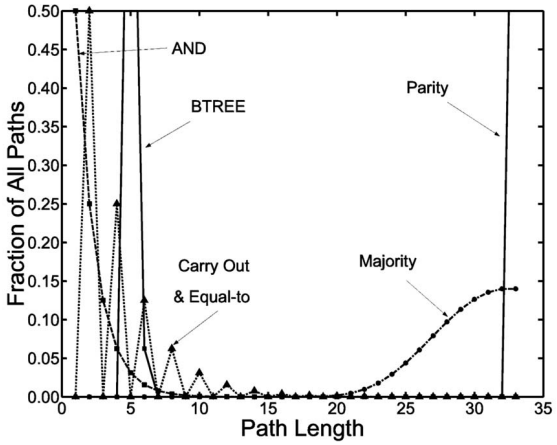

Fig. 8. Distributions of path lengths for various 33-variable functions.

Definition 3. *The Achilles' heel function is*

$$f(x_1, x_2, \ldots, x_n) = x_1x_2 \vee x_3x_4 \vee \ldots \vee x_{n-1}x_n, \tag{3}$$

where n is an even positive integer.

First, we consider the Achilles' heel function using the good order (top to bottom) x_1, x_2, x_3, x_4, ..., x_{n-1}, and x_n. For this case, the number of nodes is n. We have

Theorem 11.

$$APL_{AchillesGoodOrder(n)} = 6 - 6\left(\frac{3}{4}\right)^{\frac{n}{2}}.$$

Proof. See the Appendix. □

As $n \to \infty$, $APL_{AchillesGoodOrder(n)} \to 6$. Like the unate cascade and comparison functions, the APL of the BDD for the Achilles' heel function (with this order) is small.

Next, we consider the APL of the BDD of the Achilles' heel function using the *bad* order (top to bottom) x_1, x_3, x_5, ... x_{n-1} x_2, x_4, x_6, ..., and x_n.

Theorem 12.

$$APL_{AchillesPoorOrder(n)} = \frac{n}{2} + 2 - 2\left(\frac{3}{4}\right)^{\frac{n}{2}}.$$

Proof. See the Appendix. □

In this case, $APL_{AchillesPoorOrder(n)}$ is approximately $\frac{n}{2}$ for large n, in contrast to a near constant 6 for the APL of a BDD with the good order x_1, x_2, x_3, x_4, ... , x_{n-1}, and x_n.

2.9 Distribution of Path Lengths

In this section, we compare the distributions of path lengths for various functions. These were produced by the generating functions used in the Appendix to compute the APL. Fig. 8 shows these distributions for five functions on $n = 33$ variables. This graph shows that the AND function has many short paths compared to the Majority and Parity functions, for example. The Carry-Out/Equal-to

and BTREE function can also be seen to have reasonably short paths. In order to show the detail of the distributions, the y axis is limited to 0.5, truncating the curves associated with the BTREE and Parity functions.

3 APL for Sets of Functions

While the values of the APL in BDDs of example functions are useful in comparing with the APL of a given function, knowing the average of the APL over sets of functions is also interesting. We ask and answer the question

What is the expected value of APL for three classes of functions—symmetric functions, symmetric threshold functions, and all functions?

Of particular interest is whether the expected value falls near the maximum, such as the parity function, or near the minimum, such as the AND function. We use $AvgAPL_S$ to denote the average of the APL over all functions belonging to the set S of functions.

3.1 Set of Symmetric Functions

It is known [23] that the BDD of any symmetric function dependent on n variables has at least one path that is the longest possible, n. We extend this by showing that $AvgAPL_{AllSym}(n)$, the average of the APL of BDDs over all symmetric functions, approaches n as n increases. Specifically,

Theorem 13.

$$AvgAPL_{AllSym}(n) = n - 1 - \frac{1}{2^n}.$$

Proof. See the Appendix. □

3.2 Set of Symmetric Threshold Functions

Recall that a symmetric threshold function is a symmetric function that is 1 iff k or more of the variables are 1. With respect to $AvgAPL_{SymThres}$, the average APL over all symmetric threshold functions, we have

Theorem 14.

$$AvgAPL_{AllSymThres}(n) = \frac{n^2 + n}{2(n+2)}.$$

Proof. See the Appendix. □

From this, it follows that

Corollary 3.

$$AvgAPL_{AllSymThres}(n) \sim \frac{n}{2}.$$

This is unusual. Namely, the set of symmetric threshold functions represents a set whose $AvgAPL$ does *not* approach n as n increases.

3.3 Set of All Functions

In the case of symmetric functions, we can ignore ordering since *any* ordering yields a minimal BDD. That is not the case for arbitrary functions. In the case of all functions, we determine an upper bound on $AvgAPL_{ALL}(n)$, the APL for all functions on n variables. Namely, we compute

TABLE 2
$AvgAPL_{all}(n)$ for $3 \leq n \leq 16$ Obtained by Averaging 1,000 Samples and from Theoretical Calculations

n	$AvgAPL_{All}(n, \Pi)$ Theoretical (5)	$AvgAPL_{All}(n)$ Experimental	# Samples
3	2.187500	2.031250	† 256
4	3.183594	2.947937	† 65536
5	4.183578	3.972250	1000
6	5.183578	4.963125	1000
7	6.183578	5.996500	1000
8	7.183578	7.024969	1000
9	8.183578	8.062383	1000
10	9.183578	9.089324	1000
11	10.183578	10.114357	1000
12	11.183578	11.132417	1000
13	12.183578	12.146977	1000
14	13.183578	13.157292	1000
15	14.183578	14.164699	1000
16	15.183578	15.169641	1000

† *Obtained by enumerating all functions.*

$AvgAPL_{All}(n, \Pi)$, the APL over all n-variable functions for *fixed ordering* Π of the variables. We have

Theorem 15.

$$AvgAPL_{All(n)} \leq AvgAPL_{All(n,\Pi)} = (n-1) + \frac{1}{2} - \sum_{i=1}^{n-1} \frac{1}{2^{2^i}}. \quad (4)$$

Proof. See the Appendix. □

For $n \to \infty$, we can write

Corollary 4.

$$AvgAPL_{All}(n) \leq AvgAPL_{All}(n, \Pi) \sim n - 1 + 0.183578. \quad (5)$$

The upper bounds expressed in (4) and (5) can be substantiated by comparing them with statistical data. The third column of Table 2 shows the APL for sets of functions on n variables for $3 \leq n \leq 16$. For $n = 3$ and 4, the average of the APL was taken over all functions. For each n in the range $5 \leq n \leq 14$, 1,000 sample functions were generated. For each function, the APL was computed by a heuristic [11] using variable sifting, yielding a minimal or near-minimal APL. Since a minimal APL is not guaranteed, the experimental data in Table 2 (third column) also represents an upper bound. However, it is likely to be a close approximation to the actual value of $AvgAPL_{All}(n)$ and we have chosen to label the column $AvgAPL_{All}(n)$. Note that the data indeed shows an APL close to n. As n increases, $AvgAPL_{All}(n)$, as approximated by 1,000 samples, approaches the upper bound in (4), which is given in the second column. That is, the second column of Table 2 shows a theoretical result, as expressed by in (4), namely, the upper bound, $AvgAPL_{All}(n, \Pi)$, derived by averaging the APL over all functions, assuming that, in all functions, the same ordering is applied to the variables.

TABLE 3
$AvgAPL_{all}(n,k)$ for $n =$ 6, 8, and 10 Variables

$k/2^n$	$AvgAPL_{all}(n,k)$			Number of
	$n=6$	$n=8$	$n=10$	Samples
0.0000	0.000000	0.000000	0.000000	1
0.0625	2.828519	4.779448	6.913753	10,000
0.1250	3.762500	5.760283	7.880319	10,000
0.1875	4.217962	6.278444	8.377202	10,000
0.2500	4.536800	6.597638	8.683376	10,000
0.3125	4.741725	6.802536	8.878707	10,000
0.3750	4.870406	6.931597	9.001665	10,000
0.4375	5.944869	7.005006	9.070214	10,000
0.5000	5.968644	7.027714	9.091589	10,000

3.4 Set of All Functions with a Specified Number of Minterms

Instead of considering all n-variable functions as a single set, as we did in the previous section, we now divide this set into subsets, according to the number true minterms. A *true minterm* of a function f is an assignment of values to *all* variables that causes f to be 1. For example, the AND function has exactly one true minterm ($x_1x_2\ldots x_n = 11\ldots1$). Functions with few true minterms tend to have simple BDDs. We seek to determine how the number of true minterms affects the APL. For most functions, the number of assignments of values to the variables that yield 0 nearly equals the number that yield 1. However, certain important functions, like the AND and OR, have a large disparity in these two numbers.

We investigated this for functions on n = 6, 8, and 10 variables. For each value of n, we randomly chose 10,000 functions in which the fraction of assignments that mapped to 1 ranged over $k/2^n = 1/16$, $1/8$, $3/16$, $1/4$, $5/16$, $3/8$, $7/16$, and $1/2$. For each set of 10,000 functions, we computed the APL using the heuristic method of [11]. As in the previous section, the use of a heuristic method means that the data represents an upper bound of $AvgAPL_{all}(n,k)$; however, the values shown are likely to be close to the exact values and we use $AvgAPL_{all}(n,k)$ to label the column. Data above $k/2^n = 1/2$ is similar since a minimal BDD for some function $\bar{f}$ can be obtained from a minimal BDD for f by interchanging the terminal nodes 0 and 1. This latter data is omitted. Table 3 shows the data. It can be seen that the APL for functions increases with the fraction of assignments mapping to 1 up to 0.5. This coincides with our intuition that functions with small k are simpler than functions with large k and thus, their BDDs have fewer nodes and smaller APL.

4 Benchmark Functions

The results of the previous section show that the average APL over all symmetric functions and over all functions is close to n, the maximum. This contrasts with the results from the section on individual functions, showing that certain commonly used functions, like the AND, OR, and carry-out functions, have small APL. Therefore, it is interesting to examine benchmark functions since these attempt to represent functions used in practical logic design. We selected 189 benchmark functions from ISCAS 85 [24] and the Logic Synthesis Workshop (LGSynth 93). Since many have multiple outputs, this actually represents 4,352 single-output functions. Using the heuristic minimizer [11], we minimized each function's APL. Then, we plotted its APL divided by n versus n, where n is the number of variables.

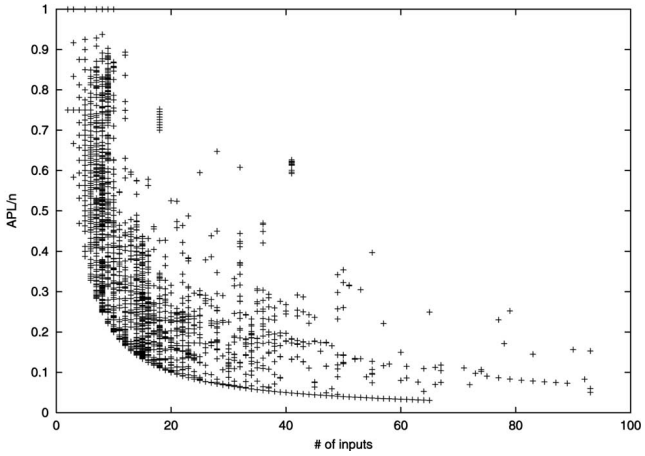

Fig. 9. $\frac{APL}{n}$ versus n for 4,352 benchmark functions.

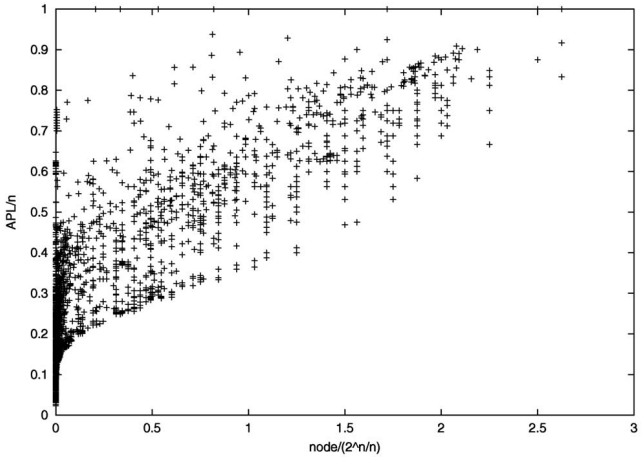

Fig. 10. $\frac{APL}{n}$ versus $\frac{N}{2^n/n}$ for 4,352 benchmark functions.

The results are shown in Fig. 9. A data point near the top corresponds to an average APL close to n. Note the general decline in APL/n as n increases. This set of benchmark functions includes many unate cascade functions and the lower bound associated with these functions is clearly seen. Contrasting the case of random functions, the values of APL/n for benchmark functions are small, especially for large n. Thus, most benchmark functions have quite different properties than randomly generated ones. The suggestion is that, for functions found in practice, the APLs of the BDDs are much smaller than n.

Fig. 10 shows this same data, except, for the horizontal axis, instead of plotting n, we plot $\frac{N}{2^n/n}$, where N is the number of nodes in the BDD. This shows that, when $\frac{N}{2^n/n}$ is large, the APL tends to be large. This shows that BDDs with more nodes tend to have larger APL. We have verified that the lower bound line is approximately defined by unate functions.

TABLE 4
Summary of the APL in BDDs of Various Functions for Large n

Function	APL
Unate Cascade (e.g. AND and OR)	$2-\frac{1}{2^{n-1}}*$
Parity	$n*$
Carry-out	$4-\frac{5}{4^n}*$
Symmetric Threshold (n,t)	$2\min\{t, n-t+1\}$
Majority	$n-\sqrt{\frac{2n}{\pi}}$
BTREE (Balanced Tree)	$\log_2 n$
Achilles' heel	$6-6(\frac{3}{4})^{\frac{n}{2}}*$
All Functions	$\leq n-0.816$
All Symmetric	$n-1-\frac{1}{2^n}$
All Symmetric Threshold	$\frac{n^2+n}{2(n+2)}*$

* *Exact value.*

5 Conclusions

Because it is related to the time of evaluation, the APL of a BDD is a useful metric in analyzing a BDD. We have shown that the APL for some functions, such as the carry-out, depends strongly on the ordering of the variables. That is, it can be as small as 4 or as large as n, for large n. We have, for the first time, contrasted and compared the APL for a set of common functions, including AND, OR, Exclusive OR, majority, unate cascade, and Greater-than-or-equal-to functions. Included in this set are the worst as well as the best APL over all n variable functions. Our approach has been to compute a distribution using generating functions from which the APL is derived. Distributions of path lengths also exhibit variability from wide, such as the majority function, to narrow, as with the parity function. Table 4 summarizes our results on APL.

We have also shown that the average of the APL over symmetric functions and over all functions is large, approaching n, the maximum as n increases. It is interesting that the average APL over all functions does not approach n exactly: rather, n less a constant. We have shown that, among all n-variable functions, the APL of the parity function is uniquely the function with maximum APL, n. We have shown empirical evidence that suggests the actual average approaches n, as n increases. In the case of symmetric threshold functions, this average approaches $\frac{n}{2}$. For benchmark functions, the APLs are usually much less than n.

A topic only briefly discussed in this paper is the correlation between minimum APL and the minimum number of nodes. This has been discussed in papers on minimization algorithms for APL [10], [11], where it has been observed that, for many benchmark functions, the BDD with the minimum APL does not correspond to the BDD with minimum node count. There is an approximate correlation. For example, [27] shows that the APL of a BDD is the sum of the node traversing probabilities and it follows that reducing the node count tends to reduce the APL.

Appendix

Theorem 1.

$$APL_{AND(n)} = 2 - \frac{1}{2^{n-1}}.$$

Proof. We begin by deriving the generating function [25] for the distribution of path lengths in the BDD of the AND function. For the AND function on n variables, there are 2^{n-1} assignments of values for which the path length is 1 ($x_1 = 0$ and two choices for each of the remaining $n-1$ variables). This contributes $2^{n-1}z^1$ to the generating function. Similarly, there are 2^{n-2} assignments of values to variables that have path length 2 and these contribute $2^{n-2}z^2$ to the generating function. The other terms are generated in a similar manner. Therefore, for the n-variable AND, the generating function for the path length distribution is

$$2^{n-1}z + 2^{n-2}z^2 + \cdots + 2^2 z^{n-2} + 2^1 z^{n-1} + 2\cdot 2^0 z^n = \frac{2^n - \frac{z^{n+1}}{2}}{1-\frac{z}{2}} + z^n - 2^n.$$

Dividing this by 2^n yields $G_{AND(n)}(z)$, a generating function for the fraction of paths of length $1, 2, 3, \ldots$.

$$G_{AND(n)}(z) = \frac{1-(\frac{z}{2})^{n+1}}{1-\frac{z}{2}} + \left(\frac{z}{2}\right)^n - 1. \tag{6}$$

The APL for the AND function, $APL_{AND(n)}$ is calculated by summing the path lengths and dividing by 2^n. This can be done by differentiating $G_{AND(n)}(z)$ with respect to z (forming a weighted sum), multiplying by z, and setting $z = 1$. Performing these steps on (6) yields the theorem. □

Theorem 2. *The unate cascade functions are uniquely those functions whose BDDs have the smallest $(2-\frac{1}{2^{n-1}})$ APL among all functions that depend on n variables.*

Proof. It is clear that $f(X)$ is a unate cascade function iff it has a BDD in which all internal nodes have one edge going to a terminal constant node. It remains to show that this BDD has the smallest APL. The proof proceeds by induction on n. For $n = 1$, a 1-variable unate function has a BDD whose APL is 1, which is the smallest of all 1-variable functions.

Assume the theorem holds for $n = m - 1$. A BDD for an m-variable function with the smallest APL has a structure in which the root node connects to a constant node by a single edge and the other connects to a BDD whose APL is the smallest for functions of $m-1$ variables. The APL is $\alpha(m) = \frac{1}{2} + \frac{1}{2}\alpha(m-1)$, where $\alpha(m-1)$ is the smallest APL among $m-1$-variable functions. Solving this recurrence relation with the initial conditions $\alpha(1) = 1$ yields $\alpha(m) = 2 - \frac{1}{2^{m-1}}$. □

Theorem 3. *The parity functions are uniquely those functions whose BDDs have the largest (n) APL among all functions that depend on n variables.*

Proof. In the BDD for one parity function, the EXOR, all paths have length n. This can be seen from Fig. 3, or it can be inferred from the fact that every variable is indispensable, i.e., every variable value is needed to determine the function value, and all paths have an arc labelled by every variable. Thus, $APL_{EXOR(n)} = n$. A similar statement is true of the other parity function, the EXNOR, which is the complement of the EXOR.

Consider any function f other than a parity function. It follows that there are two assignments A_1 and A_2 of values to the variables that differ in exactly one variable, say x_j, such that the function has the *same* value for both assignments. Consider a BDD with x_j at the bottom. It follows that a path from the root node to a terminal node corresponding to either assignment A_1 or A_2 has $n-1$ or fewer edges since x_j need not be evaluated. Since at least one path has length $n-1$ or less, the APL of f is strictly less than n. □

Theorem 6. *The smallest APL in a BDD for the Greater-than-or-equal-to function is* $4-\frac{5}{2^b}$ *and is achieved when variables are ordered in descending significance.*

Proof. Let $X=\{x_{b-1},x_{b-2},\ldots,x_0\}$ and

$$Y=\{y_{b-1},y_{b-2},\ldots,y_0\}$$

be the input variables of the Greater-than-or-equal-to function. Let $x=x_{b-1}2^{b-1}+x_{b-2}2^{b-2}+\ldots+x_02^0$ and $y=y_{b-1}2^{b-1}+y_{b-2}2^{b-2}+\ldots+y_02^0$ be the standard binary representations of the X and Y variables, respectively. Then, the Greater-than-or-equal-to function $f_{x\geq y}(X,Y)=1$ iff $x\geq y$.

The proof proceeds by induction on b, where $b=|X|=|Y|$. For $b=1$, the Greater-than-or-equal-to function is $f_{x\geq y}=x_0\vee\bar{y}_0$, which is a unate cascade function, such that $APL=\frac{3}{2}$. Since $4-\frac{5}{2^b}=\frac{3}{2}$, for $b=1$, the theorem holds for $b=1$. Assume it holds for $f_{x'\geq y'}(X',Y')$, where $|X'|=|Y'|=b-1$, and consider the case of $f_{x\geq y}(X,Y)$, where $|X|=|Y|=b$. The proof proceeds by showing that the minimum APL occurs when the two most significant bits, x_{b-1} and y_{b-1}, are the top two variables in the BDD. This yields a BDD containing the BDD of a $b-1$ bit Greater-than-or-equal-to function. In turn, its minimum APL is achieved when the two most significant bits, x_{b-2} and y_{b-2}, are at the top, etc. The theorem follows from this.

For all orderings of variables in the BDD of the Greater-than-or-equal-to function, there is no edge from the root node to a terminal node since the determination of whether $x\geq y$ or not requires at least two variables. Therefore, for all orderings of the BDD of the Greater-than-or-equal-to function, all paths have length at least 2 and the BDD has the structure shown in Fig. 11. Here, $z_1,z_2\in\{x_{b-1},x_{b-2},\ldots,x_0,y_{b-1},y_{b-2},\ldots,y_0\}$ and $z_1\neq z_2$.

In this structure, the four triangles represent the BDDs of subfunctions of the Greater-than-or-equal-to function corresponding to $z_1z_2=00,\ 01,\ 10,\ 11$. We consider three cases: 1) $\{z_1,z_2\}=\{x_{b-1},y_{b-1}\}$, 2) $\{z_1,z_2\}=\{x_k,y_k\}$, $k\neq b-1$, and 3) all other choices for z_1 and z_2. In Case 1, B_{01} and B_{10} are null; the edges going to their root nodes go to terminal nodes, 0 and 1, respectively. Further, B_{00} and B_{11} are identical and represent the BDD of a Greater-than-or-equal-to function with $b-1$ bit pairs. By induction, an optimum ordering of that BDD yields an APL of $4-\frac{5}{2^{b-1}}$. Thus, the APL of the BDD with $\{z_1,z_2\}=\{x_{b-1},y_{b-1}\}$ is

$$\frac{1}{2}2+\frac{1}{2}\left(2+4-\frac{5}{2^{b-1}}\right)=4-\frac{5}{2^b}. \tag{7}$$

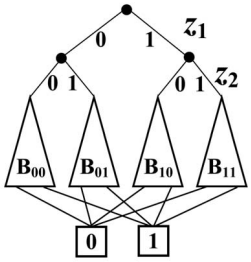

Fig. 11. BDD Structure of the Comparison Function.

In Case 2, B_{00} and B_{11} are BDDs for Greater-than-or-equal-to functions on $b-1$ variables. B_{01} and B_{10} represent BDDs that depend on only variables of higher significance than i. The derivation of the APL, for this case, is similar to that of Case 1 and yields a value greater than that of Case 1.

Now, consider Case 3, where $\{z_1,z_2\}\neq\{x_i,y_i\}$. We show that, in Fig. 11, all sub-BDDs, B_{00}, B_{01}, B_{10}, and B_{11} depend on all $2b-2$ variables. Thus, a lower bound on the APL for this case is

$$\begin{aligned}2+\frac{1}{4}LB(n-2)+\frac{1}{4}LB(n-2)+\frac{1}{4}LB(n-2)\\ +\frac{1}{4}LB(n-2)=2+LB(n-2),\end{aligned} \tag{8}$$

where $LB(n-2)$ is a lower bound on the APL of a function of $n-2$ variables. From Theorem 2, this is $2-\frac{1}{2^{n-3}}$. Thus, the APL for this case is

$$4-\frac{1}{2^{2b-3}}, \tag{9}$$

which is also larger than the APL for Case 1. Therefore, the smallest APL occurs with x_{b-1} and y_{b-1} as the top two variables.

We now complete the proof by showing that each B_{00}, B_{01}, B_{10}, and B_{11} realizes a function dependent on $n-2=2(b-1)$ variables. We do it by contradiction. That is, on the contrary, suppose that some variable, say x_i (y_i) is not tested in $B_j\in\{B_{00},B_{01},B_{10},B_{11}\}$. Then, consider an assignment of values such that $y_i=1$ ($x_i=0$) and all other values are 0(1). Then, whether $x\geq y$ depends on x_i (y_i), the missing variable. It follows that all variables are tested in B_j. □

Theorem 7.

$$APL_{S\_THRES(n,t)}=2k-\sum_{j=1}^{k}\frac{\binom{n-j}{k-j}}{2^{n-j}}j, \tag{10}$$

where $k=\min\{t,n-t+1\}$.

Proof. The BDD of a symmetric threshold function is a rectangle of $t\times(n-t+1)$ nodes, one corner of which serves as the root node and the opposite corner is just above the two terminal nodes. For example, Fig. 3a shows the case of an $n\times 1$ rectangle, representing a symmetric threshold function where $t=n$ (AND function). Also, Fig. 6 shows the case of a 3×3 rectangle

representing a symmetric threshold function with $t = \frac{n+1}{2}$ for $n = 5$ (majority function). There are $n - t + 1$ arcs emerging from the right side of the rectangle going to nonterminal node 1. And, there are t arcs emerging from the left side of the rectangle going to nonterminal node 0. Since all paths from the root node to a nonterminal node pass through one of these arcs, we can calculate the APL by summing the fraction of paths associated with each arc multiplied by the path length associated with that arc. Doing this yields the generating function for the distribution of path lengths for the symmetric threshold function with threshold t as

$$G_{S\_THRES(n,t)}(z) = \sum_{j=k}^{n} \left[\binom{j-1}{k-1} + \binom{j-1}{n-k} \right] \left(\frac{z}{2}\right)^j,$$

where $k = \min\{t, n - t + 1\}$.

Differentiating $G_{S\_THRES(n,t)}(z)$ with respect to z, multiplying by z, and setting z to 1 yields a weighted sum which, divided by 2^n, yields the APL given in (10).□

Theorem 8.

$$APL_{MAJ(n)} = n - \frac{n+1}{2^n} \binom{n}{\frac{n-1}{2}} + 1, \tag{11}$$

where n is odd.

Proof. The BDD of the majority function is a square of nodes with one corner at the root node and the other corner just above the two terminal nodes. Fig. 6 shows the BDD of the five variable majority function. There are $t + 1$ different path lengths. The shortest occurs when the first t variables are all 1. In this case, the majority function is 1, regardless of the values of the remaining $t - 1$ variables. There are 2^{t-1} assignments to these variables, each corresponding to a path of length t. Thus, the contribution to the path length distribution is expressed as $2^{t-1}z^t$. The next shortest paths correspond to assignments of values to the variables in which there are t 1s and one 0. This yields a contribution to the distribution of $2^{t-2}\binom{t}{1}z^{t+1}$. The contribution associated with the path length assignments in which there are t 1s and two 0s is $2^{t+1}\binom{t}{2}z^{t+2}$, etc. Thus, the generating function for the path length distribution is

$$2\left(2^{t-1}z^t + 2^{t-2}\binom{t}{1}z^{t+1} + 2^{t-3}\binom{t+1}{2} z^{t+2} + \ldots \quad + \binom{2t-2}{t-1}z^{2t-1}\right).$$

The factor of 2 occurs because, for every path from the root node to terminal node 1 associated with t 1s and i 0s, there is a path from the root node to terminal node 0 associated with t 0s and i 1s. Rearranging this expression yields

$$G_{MAJ(n)}(z) = 2^t z^t \sum_{i=0}^{t-1} \binom{t-1+i}{t-1} \left(\frac{z}{2}\right)^i, \tag{12}$$

where $n = 2t - 1$, for $t = 1, 2, \ldots$.

We compute $APL_{MAJ(n)}$ by summing the path lengths over all assignments of values to the variables and dividing by 2^n, the number of assignments. If all path lengths are the maximum value, n, the sum of path lengths over all assignments is $n2^n$. Thus,

$$APL_{MAJ(n)} = \frac{n2^n - R(n)}{2^n}, \tag{13}$$

where $R(n)$ is a reduction in the maximum weighted sum caused by paths that do not reach the maximum length. In the case of the majority function, all such paths extend from some lower-middle node to a terminal node. For example, if the first (top) $m + 1$ variables are 1, where $m = \frac{n-1}{2}$, then the majority function is 1 regardless of the value of the remaining m variables, which can be chosen in any of 2^m ways. In the BDD of the majority function, this corresponds to 2^m paths, each contributing a reduction of m to $R(n)$, for a total reduction of $2^m m$. In the case where the first $m + 2$ variables have $m + 1$ 1s and one 0, the majority function is 1 regardless of the value of the remaining $m - 1$ variables, which can be chosen in 2^{m-1} ways. In the BDD of the majority function, this corresponds to 2^{m-1} paths, each contributing a reduction of $m - 1$ to $R(n)$. Since there are $\binom{m+1}{1}$ ways to choose where the 0 occurs, the total contribution to the reduction $R(n)$ in this case is $\binom{m+1}{1}2^{m-1}(m-1)$. In general, where the first $m + 1 + i$ variables have exactly i 0s, the contribution to $R(n)$ is $\binom{m+i}{i}2^{m-i}(m-i)$ and we can write

$$R(n) = 2\sum_{i=0}^{m} \binom{m+i}{i} 2^{m-i}(m-i), \tag{14}$$

where $m = \frac{n-1}{2}$. The factor 2 occurs because there are corresponding paths that go to terminal node 0.

From Riordan [26], the sum in (14) represents the probability distribution associated with Banach's matchbox problem, which has a closed form solution. Namely,

$$\sum_{i=0}^{m} \binom{m+i}{i} 2^{m-i}(m-i) = (m+1)\binom{2m+1}{m}2^{-2m} - 1. \tag{15}$$

Combining (13), (14), and (15) yields (11). □

Theorem 9.

$$APL_{BTREE(n)} = \lfloor \log_2(n+1) \rfloor + \frac{(n+1) - 2^{\lfloor \log_2(n+1) \rfloor}}{\lfloor \log_2(n+1) \rfloor}. \tag{16}$$

Proof. In the BDD of the BTREE function on n variables, when $n = 2^{\alpha(n)} - 1$ for $\alpha(n) = 1, 2, \ldots$, all 2^n assignments of values to the variables correspond to a path of length $\alpha(n) = \lfloor \log_2 n \rfloor$. For other values of n, namely, for $2^{\alpha(n)} - 1 < n < 2^{\alpha(n)+1} - 1$, there are $(2^{\alpha(n)} - E)2^{2^\alpha + E - 1 - \alpha(n)}$ assignments of values to the variables that correspond to a path of length $\alpha(n)$ and $E2^{2^\alpha + E - 1 - \alpha(n)}$ to a path of length $\alpha(n) + 1$. From this, the generating function for the distribution of path lengths in the BTREE function is given as

$$G_{BTREE(n)}(z) = [1 - (n - 2^{\lfloor \log_2 n \rfloor} + 1)2^{-\lfloor \log_2 n \rfloor}]z^{\lfloor \log_2 n \rfloor} + [n - 2^{\lfloor \log_2 n \rfloor} + 1]2^{-\lfloor \log_2 n \rfloor} z^{\lfloor \log_2 n \rfloor + 1}. \quad (17)$$

Differentiating $G_{BTREE(n)}(z)$ with respect to z, multiplying by z, and setting z to 1, yields a weighted sum which divided by 2^n yields the APL given in (16). □

Theorem 11.

$$APL_{AchillesGoodOrder(n)} = 6 - 6\left(\frac{3}{4}\right)^{\frac{n}{2}}.$$

Proof. The BDD for the Achilles' Heel function in the good order is a cascade of sections each containing a BDD of the two variable AND. $A(Z) = \frac{1}{4}z^2 + \frac{1}{2}z$ is the distribution of paths from the root node of this section to the logic 0 side. The generating function for the distribution of paths from the root node of the Achilles' Heel good order BDD is just $A(z)^{\frac{n}{2}}$, as each $A(z)$ enumerates the path through the corresponding section. Similarly, $\frac{1}{4}z^2 + \frac{1}{4}z^2A(z) + \frac{1}{4}z^2A(z)^2 + \ldots + \frac{1}{4}z^2A(z)^{\frac{n}{2}-1}$, where $\frac{1}{4}z^2$ expresses the path probability beginning from the root node of a section and the various $A(z)^i$ terms represent the ways to go to a root of a section. Adding the two generating functions yields the generating function for the distribution of path lengths for both the 0 and 1 nonterminal nodes

$$G_{AchillesGoodOrder(n)}(z) = \frac{1}{4}z^2\frac{1 - (\frac{1}{4}z^2 + \frac{1}{2}z)^{n/2}}{1 - (\frac{1}{4}z^2 + \frac{1}{2}z)} + \left(\frac{1}{4}z^2 + \frac{1}{2}z\right)^{n/2}. \quad (18)$$

We derive the APL by differentiating

$$G_{AchillesGoodOrder(n)}(z)$$

with respect to z, multiplying by z, and substituting $1 \rightarrow z$. Doing this yields the theorem. □

Theorem 12.

$$APL_{AchillesPoorOrder(n)} = \frac{n}{2} + 2 - 2\left(\frac{3}{4}\right)^{\frac{n}{2}}.$$

Proof. The BDD for the order x_1, x_3, x_5, … x_{n-1} x_2, x_4, …, and x_n consists of a complete balanced tree labeled by x_1, x_3, x_5, … x_{n-1} in which all paths have lengths $\frac{n}{2}$. Each leaf in this tree is the root node of a *different* sub-BDD of a function that is the OR. For example, in the case where $x_1 = x_3 = x_5 = \ldots = x_{n-1} = 1$, the sub-BDD is for the OR function $x_2 \vee x_4 \vee x_6 \vee \ldots \vee x_n$. The APL for paths beginning with $x_1 = x_3 = x_5 = \ldots = x_{n-1} = 1$ is $\frac{n}{2} + 2 - \frac{1}{2^{\frac{n}{2}-1}}$ since $2 - \frac{1}{2^{\frac{n}{2}-1}}$ is the APL for an $\frac{n}{2}$-variable OR function. For other values of x_1, x_3, …, and x_{n-1}, the sub-BDD is that of an OR function on α variables, where α is the number of 1s among x_1, x_3, …, and x_{n-1}. It follows that

$$APL_{AchillesPoorOrder(n)} = \frac{n}{2} + \frac{1}{2^{\frac{n}{2}}}\sum_{i=0}^{\frac{n}{2}}\binom{\frac{n}{2}}{i}\left(2 - \frac{1}{2^{i-1}}\right).$$

Distributing the sum over the two terms in $\left(2 - \frac{1}{2^{i-1}}\right)$ and expressing each in closed form yields the theorem. □

Theorem 13.

$$AvgAPL_{AllSym}(n) = n - 1 - \frac{1}{2^n}. \quad (19)$$

Proof. A symmetric function, $f(x_1, x_2, ..., x_n)$ has the Shannon decomposition, $f = \bar{x}_1 f_0 \vee x_1 f_1$, where $f_0 = f|_{x_1=0}$ and $f_1 = f|_{x_1=1}$. From this realization, we have

$$AvgAPL_{AllSym}(n) = \frac{1}{2}(1 + AvgAPL_{AllSym}(n-1)) + \frac{1}{2}(1 + AvgAPL_{AllSym}(n-1)) - \frac{2}{2^{n+1}}.$$

The factor of $\frac{1}{2}$ corresponds to the fact that one-half of the paths begin in 0 and one-half in 1. The $1+$ term corresponds to the edge from the root node to the BDDs realizing f_0 and f_1. The term $-\frac{2}{2^{n+1}}$ corresponds to a deduction associated with choosing both f_0 and f_1 as a constant 0 or 1 function. These contribute $\frac{2}{2^{n+1}}$, but they should contribute 0. Solving this recurrence relation yields (19). □

Theorem 14.

$$AvgAPL_{AllSymThres}(n) = \frac{n^2 + n}{2(n+2)}.$$

Proof. There are $n + 2$ symmetric threshold functions on n variables, including the constant 0 and 1 functions, corresponding to thresholds 0 and $n + 1$, respectively. If all paths in each have maximum length, then the weighted sum over all paths is $2^n n$ as there are 2^n paths, each of length n. However, many paths are truncated and the weighted sum is actually $2^n n(n+2) - \alpha(n)$, where $\alpha(n)$ is a *reduction*. The reduction occurs because paths are truncated when enough is known to completely determine that the threshold is exceeded or not exceeded regardless of the remaining variable values. For the two extreme thresholds, the function is a constant 0 or 1 and the reduction is $2^n n$, completely eliminating the BDD. This explains the $2^{n+1}n$ term in

$$\alpha(n) = 2^{n+1}n + 2\sum_{k=0}^{n-1} 2^k k \sum_{j=0}^{n-1-k}\binom{n-1-k}{j}. \quad (20)$$

In (20), the sum over k counts the reduction when the threshold is not exceeded and, thus, the factor of 2 accommodates the case when the threshold is exceeded as well since the two cases are symmetrical. The sum over k enumerates the cases where there are k variables below the present level and none are needed to determine the function value (it is 0, regardless of these variable values). There are 2^k ways to choose these values and each choice reduces the path length

by k. The sum over j enumerates the ways 0s can be chosen for variables in the top of the BDD such that the threshold is not exceeded. $j = 0$ corresponds to the threshold $t = k + 1$. In this case, there is $\binom{n-1-k}{0} = \binom{n-t}{0} = 1$ way to choose none among the $n - 1 - k = n - t$ top variables to be 1. One more 0 results in $n - t + 1$ 0s among the top variables and guarantees the function is 0. $j = 1$ corresponds to the threshold $t = k + 2$. In this case, there are $\binom{n-1-k}{1} = \binom{n-t+1}{1} = n - t + 1$ ways to choose one among the top $n - t + 1$ variables to be 1. One more 0 results in $n - t + 1$ 0s among the top variables and guarantees the function is 0. Similarly, this argument applies to values of j up to $n - 1 - k$.

The summation over j is just 2^{n-1-k}. Substituting into (20) and rearranging yields

$$\alpha(n) = 2^{n+1}n + 2^n \sum_{i=0}^{n-1} i. \tag{21}$$

The summation is $\frac{n(n-1)}{2}$. Substituting yields

$$\alpha(n) = (n^2 + 3n)2^{n-1}. \tag{22}$$

We have

$$AvgAPL_{AllSymThres}(n) = \frac{2^n n(n+2) - \alpha(n)}{2^n(n+2)}. \tag{23}$$

Substituting (22) into (23) yields the theorem. □

Theorem 15.

$$AvgAPL_{All}(n) \leq AvgAPL_{All}(n, \Pi) = (n-1) + \frac{1}{2} - \sum_{i=1}^{n-1} \frac{1}{2^{2^i}}. \tag{24}$$

Proof. A general function, $f(x_1, x_2, ..., x_n)$ has the Shannon decomposition, $f = \bar{x}_1 f_0 \vee x_1 f_1$, where $f_0 = f|_{x_1=0}$ and $f_1 = f|_{x_1=1}$. From this realization, we have

$$AvgAPL_{All}(n, \Pi) = \frac{1}{2}(1 + AvgAPL_{All}(n-1, \Pi)) + \frac{1}{2}(1 + AvgAPL_{All}(n-1, \Pi)) - \frac{1}{2^{2^{n-1}}}.$$

Here, the $\frac{1}{2^{2^{n-1}}}$ term is a deduction for the case where both subfunctions of the Shannon decomposition are the same. For each such subfunction, the root node and the two arcs incident to it are redundant and the APL is one more than it should be. Solving this recurrence relation yields (24). □

ACKNOWLEDGMENTS

This paper is an extended version of T. Sasao, J.T. Butler, and M. Matsuura, "Average Path Length as a Paradigm for the Fast Evaluation of Functions Represented by Binary Decision Diagrams," which appeared in the *Proceedings of the First International Symposium on New Paradigm VLSI Computing*, pages 31-36, 12-14 December 2002. This research is partly supported by the Grant in Aid for Scientific Research of the Japan Society for the Promotion of Science (JSPS), funds from the Ministry of Education, Culture, Sports, Science, and Technology (MEXT) via a Kitakyushu Innovative Cluster Project, and NSA Contract RM - A-54.

REFERENCES

[1] C. Lee, "Representation of Switching Circuits by Binary-Decision Diagrams," *Bell System Technical J.*, vol. 19, pp. 685-999, July 1959.
[2] R.E. Bryant, "Graph-Based Algorithms for Boolean Function Manipulation," *IEEE Trans. Computers*, vol. 35, no. 8, pp. 677-691, Aug. 1986.
[3] N. Ishuira, H. Sawada, and S. Yajima, "Minimization of Binary Decision Diagrams Based on Exchanges of Variables," *Proc. IEEE Int'l Conf. Computer-Aided Design*, pp. 472-475, Nov. 1991.
[4] R. Rudell, "Dynamic Variable Ordering for Ordered Binary Decision Diagrams," *Proc. IEEE Int'l Conf. Computer-Aided Design*, pp. 42-47, Nov. 1993.
[5] R. Drechsler, N. Drechsler, and W. Gunther, "Fast Exact Minimization of BDDs," *IEEE Trans. Computer-Aided Design*, vol. 19, pp. 384-389, Mar. 2000.
[6] S. Nagayama and T. Sasao, "On the Minimization of the Longest Path Length for Decision Diagrams," *Proc. Int'l Workshop Logic Synthesis*, pp. 28-35, June 2004.
[7] P. Ashar and S. Malik, "Fast Functional Simulation Using Branching Programs," *Proc. IEEE Int'l Conf. Computer-Aided Design*, pp. 308-412, Nov. 1995.
[8] P.C. McGeer, K.L. McMillan, A. Saldanha, and A.L. Sangiovanni-Vincentelli, "Fast Discrete Function Evaluation Using Decision Diagrams," *Proc. Int'l Conf. Computer-Aided Design*, pp. 402-407, Nov. 1995.
[9] T. Sasao, Y. Iguchi, and M. Matsuura, "Comparison of Decision Diagrams for Multiple-Output Logic Functions," *Proc. 11th Int'l Workshop Logic Synthesis*, pp. 379-384, June 2002.
[10] Y.Y. Liu, K.H. Wang, T.T. Hwang, and C. Liu, "Binary Decision Diagram with Minimum Expected Path Length," *Proc. Design Automation and Test in Europe (DATE2001)*, pp. 1-5, Mar. 2001.
[11] S. Nagayama, A. Mischenko, T. Sasao, and J.T. Butler, "Minimization of Average Path Length in BDDs by Variable Reordering," *Proc. 12th Int'l Workshop Logic Synthesis*, pp. 207-213, May 2003.
[12] Y. Iguchi, T Sasao, and M. Matsuura, "Evaluation of Multiple-Output Logic Functions Using Decision Diagrams," *Proc. ASP-DAC (Asia and South Pacific Design Automation Conf.*, pp. 312-315, Jan. 2003.
[13] J. Butler and T. Sasao, "On the Average Path Length in Decision Diagrams of Multiple-Valued Functions," *Proc. 23rd Int'l Symp. Multiple-Valued Logic*, pp. 383-390, May 2003.
[14] D.A. Bell, "Decision Trees, Tables, and Lattices," *Pattern Recognition: Ideas in Practice*, B.G. Batchelor, ed., chapter 5, New York: Plenum Press, 1978.
[15] D.E. Knuth, "Mathematical Analysis of Algorithms," *Proc. IFIP Congress 71*, vol. 1, pp. 135-143, 1971.
[16] M. Hanani, "An Optimal Evaluation of Boolean Expressions in an Online Query System," *Comm. ACM*, vol. 20, pp. 344-347, May 1977.
[17] W. Qin and S. Malik, "Automated Synthesis of Efficient Binary Decoders for Retargetable Software Toolkits," *Design Automation Conf.*, pp. 764-769, June 2003.
[18] B.N.E. Moret, "Decision Trees and Diagrams," *Computing Surveys*, vol. 14, pp. 593-623, Dec. 1982.
[19] S.K. Murthy, "Automatic Construction of Decision Trees from Data: A Multi-Disciplinary Survey," *Data Mining and Knowledge Discovery*, vol. 2, pp. 345-389 Oct. 1998.
[20] Y. Breithart and S. Gal, "Analysis of Algorithms of the Evaluation of Monotonic Boolean Functions," *IEEE Trans. Computers*, vol. 27, no. 11, pp. 1083-1087, Nov. 1978.
[21] T. Sasao and K. Kinoshita, "On the Number of Fanout-Free Functions and Unate Cascade Function," *IEEE Trans. Computers*, vol. 28, no. 9, pp. 682-685, Sep. 1979.
[22] T. Sasao, *Switching Theory for Logic Synthesis*. New York: Kluwer Academic, 1999.

[23] B.N.E. Moret, M.G. Thomason, and R.C. Gonzalez, "Symmetric and Threshold Boolean Functions Are Exhaustive," *IEEE Trans. Computers,* vol. 32, no. 12, pp. 1211-1212, Dec. 1983.
[24] F. Brglez and H. Fujiwara, "A Neutral Netlist of 10 Combinational Benchmark Circuits," *Proc. IEEE Int'l Symp. Circuits and Systems,* pp. 695-698, 1985.
[25] C.L. Liu, *Introduction to Combinatorics.* New York: McGraw-Hill, 1968.
[26] J. Riordan, *Combinatorial Identities,* p. 31. New York: John Wiley & Sons, 1968.
[27] T. Sasao, Y. Iguchi, and M. Matsuura, "Comparison of Decision Diagrams for Multiple-Output Logic Functions," *Proc. Int'l Workshop Logic and Synthesis,* pp. 379-384, June 2002.
[28] M. Tachibana, "Heuristic Algorithms for FBDD Node Minimization with Application to Pass-Transistor-Logic and DCVS Synthesis," *Proc. Workshop Synthesis And System Integration of MIxed Technologies (SASIMI),* pp. 96-101, Nov. 1996.
[29] K. Yano, Y. Sasaki, K. Rikino, and K. Seki, "Top-Down Pass-Transistor Logic Design," *IEEE J. Solid State Circuits,* vol. 31, pp. 792-803. June 1996.
[30] C. Scholl and B. Becker, "On the Generation of Multiplexer Circuits for Pass Transistor Logic," *Proc. Design Automation and Test in Europe,* pp. 372-378, 2000.
[31] T.H. Liu, M.K. Ganai, A. Aziz, and J.L. Burns, "Performance Driven Synthesis for Pass Transistor Logic," *Proc. Int'l Workshop Logic Synthesis,* pp. 255-259, 1998.
[32] C. Yang and M. Ciesielski, "BDS: A BDD-Based Logic Optimization System," *IEEE Trans. CAD,* vol. 27, no. 7, pp. 866-876, 2002.
[33] R. Ebendt, W. Gunther, and R. Dreschler, "Minimization of Expected Path Length in BDDs Based on Local Changes," *Proc. ASP-DAC (Asia and South Pacific Design Automation Conf.,* pp. 866-871, Jan. 2004.

Jon T. Butler (S'67-M'67-SM'82-F'89) received the BEE and MEngr degrees from Rensselaer Polytechnic Institute, Troy, New York, in 1966 and 1967, respectively. He received the PhD degree from The Ohio State University, Columbus, in 1973. Since 1987, he has been a professor at the Naval Postgraduate School, Monterey, California. From 1974 to 1987, he was at Northwestern University, Evanston, Illinois. During that time, he served two periods of leave at the Naval Postgraduate School, first as a National Research Council Senior Postdoctoral Associate (1980-1981) and second as the NAVALEX Chair Professor (1985-1987). He served one period of leave as a foreign visiting professor at the Kyushu Institute of Technology, Iizuka, Japan. His research interests include logic optimization and multiple-valued logic. He has served on the editorial boards of the *IEEE Transactions on Computers*, *Computer*, and IEEE Computer Society Press. He has served as the editor-in-chief of *Computer* and IEEE Computer Society Press. He received the Award of Excellence, the Outstanding Contributed Paper Award, and a Distinctive Contributed Paper Award for papers presented at the International Symposium on Multiple-Valued Logic. He received the Distinguished Service Award, two Meritorious Service Awards, and nine Certificates of Appreciation for service to the IEEE Computer Society. He is a fellow of the IEEE.

Tsutomu Sasao (S'72-M'77-SM'90-F'94) received the BE, ME, and PhD degrees in electronics engineering from Osaka University, Osaka, Japan, in 1972, 1974, and 1977, respectively. He has held faculty/research positions at Osaka University, Japan, the IBM T.J. Watson Research Center, Yorktown Heights, New York, and the Naval Postgraduate School, Monterey, California. He has served as the director of the Center for Microelectronic Systems at the Kyushu Institute of Technology, Iizuka, Japan, where he is currently a professor in the Department of Computer Science and Electronics. His research areas include logic design and switching theory, representations of logic functions, and multiple-valued logic. He has published more than eight books on logic design including, *Logic Synthesis and Optimization*, *Representation of Discrete Functions*, *Switching Theory for Logic Synthesis*, and *Logic Synthesis and Verification* (Kluwer Academic, 1993, 1996, 1999, 2002, respectively). He has served as program chairman for the IEEE International Symposium on Multiple-Valued Logic (ISMVL) many times. Also, he was the symposium chairman of the 28th ISMVL held in Fukuoka, Japan, in 1998. He received the NIWA Memorial Award in 1979, Takeda Techno-Entrepreneurship Award in 2001, and Distinctive Contribution Awards from the IEEE Computer Society MVL-TC for papers presented at ISMVL in 1987, 1996, 2003, respectively. He has served as an associate editor of the *IEEE Transactions on Computers*. He is a fellow of the IEEE.

Munehiro Matsuura studied at the Kyushu Institute of Technology from 1983 to 1989. He received the BE degree in natural sciences from the University of the Air, Japan, in 2003. He has been working as a technical assistant at the Kyushu Institute of Technology since 1991. He has implemented several logic design algorithms under the direction of Professor Tsutomu Sasao. His interests include decision diagrams and exclusive-OR-based circuit design.